

Package: htna (via r-universe)

October 8, 2026

Title Heterogeneous Transition Network Analysis

Version 0.3.1

Description Implements the Heterogeneous Transition Network Analysis (HTNA) method described by López-Pernas et al. (2026) <[doi:10.1002/jcal.70285](https://doi.org/10.1002/jcal.70285)>. The method is an extension of transition network analysis (TNA) where actions or events belong to two or more distinct actor types (e.g. Human and AI), preserving the actor type partition on the resulting network. Provides a thin, focused API on top of the 'Nestimate' estimation engine and the 'cograph' rendering engine, so downstream bootstrap, permutation, reliability, centrality, and plotting functions treat each actor's codes as a distinct node group.

License MIT + file LICENSE

URL <https://sonsoles.me/htna/>

BugReports <https://github.com/sonsoleslp/htna/issues>

Depends R (>= 4.1.0)

Imports Nestimate (>= 0.8.0), cograph, igraph

Suggests codyna, ggplot2, janitor, knitr, rmarkdown, gridExtra, testthat (>= 3.0.0), tna

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

LazyData true

Config/pak/sysreqs libglpk-dev libxml2-dev

Repository <https://sonsoleslp.r-universe.dev>

Date/Publication 2026-07-28 13:05:04 UTC

RemoteUrl <https://github.com/sonsoleslp/htna>

RemoteRef HEAD

RemoteSha c9dc2944f7483079e3338d20b7b9c7a64200612e

Contents

ai_simplified	3
as.igraph.htna	4
association_rules_htna	5
bayes_compare_htna	6
bootstrap	7
bootstrap_htna	7
build_htna	8
casedrop_reliability_htna	11
centralities_htna	12
centrality_stability_htna	13
certainty_htna	15
compare_htna	16
deprune_htna	17
edge_betweenness_htna	18
extract_meta_paths	19
frequencies_htna	21
htna_palette	22
human_ai	22
human_ai_codebook	24
human_simplified	24
markov_order_test_htna	25
mosaic_plot_htna	26
permutation_htna	27
plot.htna_comparison_group	28
plot.htna_stability	29
plot.htna_stability_group	29
plot_centralities	30
plot_edge_betweenness_htna	31
plot_frequencies_htna	32
plot_htna	33
plot_htna_bootstrap	34
plot_htna_diff	35
plot_sequences	37
print.htna_stability	37
print.htna_stability_group	38
prune_htna	39
pruning_details_htna	40
reliability_htna	41
reprune_htna	42
sequence_compare_htna	43
sequence_plot_htna	44

ai_simplified

3

state_distribution_htna

45

state_frequencies_htna

46

summary.htna

47

Index

49

ai_simplified	<i>Simplified AI Interaction Sequences (per-actor frame)</i>
---------------	--

Description

The AI-only slice of [human_ai](#), in a long-format data frame ready to feed the named-list form of [build_htna\(\)](#). Codes have already been collapsed into the simplified AI alphabet; see [human_ai](#) for the remapping rules.

Usage

ai_simplified

Format

A data frame with 8551 rows and 10 columns. Schema matches [human_ai](#) minus the phase column; every value in actor_type is "AI".

Source

Derived from `Nestimate::ai_long`; see `data-raw/human_ai.R` for the build script.

See Also

[human_ai](#), [human_simplified](#), [human_ai_codebook](#).

Examples

```
data(human_simplified, ai_simplified)
net <- build_htna(list(Human = human_simplified, AI = ai_simplified))
plot_htna(net)
```

as.igraph.htna

*Convert an HTNA Network to igraph***Description**

Converts an htna network to an [igraph::igraph](#) object while preserving its node-to-actor partition. The actor type is stored as the vertex attribute actor_type, and the canonical actor ordering is stored as the graph attribute actor_levels.

Usage

```
## S3 method for class 'htna'
as.igraph(x, mode = NULL, weighted = TRUE, diag = TRUE, ...)

## S3 method for class 'htna_group'
as.igraph(x, mode = NULL, weighted = TRUE, diag = TRUE, ...)
```

Arguments

x	An htna network or htna_group.
mode	Character igraph adjacency mode. The default, NULL, uses "directed" when x\$directed is true and "undirected" otherwise.
weighted	Passed to igraph::graph_from_adjacency_matrix() . Default TRUE.
diag	Include diagonal/self-loop entries? Default TRUE.
...	Additional arguments passed to igraph::graph_from_adjacency_matrix() .

Details

Passing an htna_group converts every cohort and returns a named list. Each graph additionally stores its cohort name in the graph attribute cohort.

Value

An igraph object for an htna, or a named list of igraph objects inheriting from htna_igraph_group for an htna_group.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
graph <- igraph::as.igraph(net)
igraph::vertex_attr(graph, "actor_type")
```

`association_rules_htna`*Association Rule Mining over Transitions*

Description

htna-named alias of `Nestimate::association_rules()`. Mines frequent itemsets and association rules over a transition-count matrix under support / confidence / lift thresholds.

Usage

```
association_rules_htna(x, ...)
```

Arguments

<code>x</code>	A transition-count matrix or an htna network. See <code>Nestimate::association_rules()</code> .
<code>...</code>	Additional arguments passed to <code>Nestimate::association_rules()</code> , such as <code>min_support</code> , <code>min_confidence</code> , <code>min_lift</code> , and <code>max_length</code> . See that function for the full, current argument list.

Details

Foundation function in the htna-aware exploratory family. Works on transition-count input produced by `frequencies_htna()` or `Nestimate::build_network()`; the rules carry over to htna networks since the underlying transition counts are the same.

Suffixed `_htna` to avoid clashing with `Nestimate::association_rules()` when both packages are loaded.

Value

A list with the discovered rules and frequent itemsets. See `Nestimate::association_rules()` for details.

See Also

`frequencies_htna()`, `mosaic_plot_htna()`.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
association_rules_htna(net, max_length = 3L)
```

bayes_compare_htna	<i>Bayesian Comparison of Two Networks</i>
--------------------	--

Description

htna-named wrapper for `Nestimate::bayes_compare()`. The Bayesian complement of `permutation_htna()`: instead of a permutation null, it places a prior on the transition structure and returns the posterior distribution of edge-weight differences between two networks.

Usage

```
bayes_compare_htna(x, y = NULL, ...)
```

Arguments

<code>x, y</code>	The two networks to compare. See <code>Nestimate::bayes_compare()</code> .
<code>...</code>	Additional arguments passed to <code>Nestimate::bayes_compare()</code> , such as <code>prior</code> , <code>draws</code> , <code>ci</code> , <code>mean_threshold</code> , <code>bound_threshold</code> , and <code>seed</code> . See that function for the full, current argument list.

Details

Works on htna networks: the actor partition (`$node_groups`, `$actor_levels`, htna class) is preserved on `result$x / result$y`, and the result carries class `net_permutation`, so `plot_htna_diff()` can render it with htna's colour and layout conventions — exactly like a `permutation_htna()` result.

Suffixed `_htna` to avoid clashing with the tna verb `compare()` and to sit alongside `permutation_htna()` in the htna API.

Value

An object of class `net_bayes` (also inheriting `netdifference` and `net_permutation`). See `Nestimate::bayes_compare()` for the full slot list.

See Also

`permutation_htna()` for the permutation-test analogue, `plot_htna_diff()` to plot the result.

Examples

```
data(human_ai)
grp <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
# Pass the grouped network as it is; all pairwise comparisons are returned.
bayes_compare_htna(grp, draws = 200, seed = 1)
```

bootstrap	<i>Bootstrap generic</i>
-----------	--------------------------

Description

S3 generic dispatched on the class of `x`. Provided so `bootstrap(net)` works directly on an htna network (see [bootstrap_htna\(\)](#)).

Usage

```
bootstrap(x, ...)
```

Arguments

<code>x</code>	An object to bootstrap.
<code>...</code>	Arguments forwarded to the method.

Value

An object whose structure is method-defined.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
bootstrap(net, iter = 50)
```

bootstrap_htna	<i>Bootstrap an HTNA Network</i>
----------------	----------------------------------

Description

Thin wrapper around [Nestimate::bootstrap_network\(\)](#) that runs the standard state-level edge bootstrap on an htna network and tags the result so it is identifiable as an htna bootstrap. The bootstrap inference itself is Nestimate's: per-edge mean, sd, p-value, significance, and confidence / credibility intervals across `iter` resamples. State-level granularity is the correct level for edge-stability analysis - aggregating up to actor pairs would smear over real edge-by-edge differences.

Usage

```
bootstrap_htna(x, ...)

## S3 method for class 'htna'
bootstrap(x, ...)
```

Arguments

x	[htna] A network built with <code>build_htna()</code> .
...	Arguments forwarded to <code>Nestimate::bootstrap_network()</code> (e.g. <code>iter</code> , <code>ci_level</code> , <code>consistency_range</code> , <code>seed</code>).

Details

What this wrapper adds is identity and downstream class continuity:

- The returned object has class "htna_bootstrap" ahead of "net_bootstrap", so `inherits(boot, "htna_bootstrap")` works.
- The actor partition (`$nodes$groups`, `$node_groups`) is restored on `boot$model` from the input network so `cograph`'s auto-detect still recognizes the heterogeneous schema downstream.
- `boot$model` is re-tagged with "htna" at the front of its class chain so the chain is `c("htna", "netobject", "cograph_network")` - identical to what `build_htna()` returns.

Value

An object of class `c("htna_bootstrap", "net_bootstrap")`. All slots produced by `Nestimate::bootstrap_network()` are preserved verbatim (`$summary`, `$mean`, `$sd`, `$p_values`, `$significant`, `$ci_lower/$ci_upper`, `$cr_lower/$cr_upper`, `$model`, `$original`, etc.).

See Also

`Nestimate::bootstrap_network()` for the underlying algorithm and slot documentation. `build_htna()` for constructing the input.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
boot <- bootstrap_htna(net, iter = 50)
inherits(boot, "htna_bootstrap")           # TRUE
inherits(boot$model, "htna")               # TRUE
head(boot$summary)                         # state-level edge stability
```

build_htna	<i>Build a Heterogeneous Transition Network (HTNA)</i>
------------	--

Description

Builds a transition network over a combined sequence of two or more actor groups (e.g. Human and AI) and preserves the actor partition on the result so downstream plotting and analysis can treat each actor's codes as a distinct node group.

Usage

```

build_htna(
  data,
  actor_type = NULL,
  actor = NULL,
  node_groups = NULL,
  action = "code",
  session = "session_id",
  order = "order_in_session",
  method = "relative",
  group = NULL,
  disambiguate = FALSE,
  source_data = NULL,
  ...
)

```

Arguments

data	Either: • A named list of long-format data frames, one per actor type (e.g. <code>list(Human = human_simplified, AI = ai_simplified)</code>). All frames must share the same column schema. • A single long-format data frame. In that case either <code>actor_type</code> (row-level actor-type IDs) or <code>node_groups</code> (node-level actor-type lookup) must be supplied. • A fitted clustering result from <code>Nestimate::build_clusters()</code>, <code>Nestimate::cluster_mmm()</code> (a <code>net_mmm</code>), or <code>tna::cluster_sequences()</code>, or an already-materialized <code>netobject_group</code>. One HTNA network is returned per sequence cluster. The clustering fit is not rerun; assignments, diagnostics, fitted MMM weights, and actor metadata are preserved. • A fitted <code>Nestimate::build_mcml()</code> object. Its node clusters become the HTNA actor groups and one full node-level network is rebuilt from the original source, including transitions between clusters.
actor_type	Character. Name of the column tagging each row's actor type / group (e.g. "Human" vs "AI") when data is a single data frame. Ignored when data is a named list or when <code>node_groups</code> is supplied.
actor	Character. Name of an optional column identifying the individual actor that performed each event (e.g. a learner / user id). Forwarded to build_network with the same semantics; orthogonal to <code>actor_type</code> , which encodes the group/type partition over codes.
node_groups	Node-to-actor-type lookup, in either of two forms: • A named list mapping actor-type labels to character vectors of code names, e.g. <code>list(Human = c("Specify", "Command"), AI = c("Plan", "Execute"))</code>. • A 2-column data frame with one column named after action (the codes) and one other column tagging each code with its actor type, e.g. <code>data.frame(code</code>

= c("Specify", "Plan"), actor_type = c("Human", "AI")). The actor-type ordering follows the column's factor levels or, for character vectors, the order of first appearance.

Use `node_groups` when data is a single long-format frame with no actor-type column and you want to declare the node-to-type partition directly. Each code in `data[[action]]` must appear in exactly one entry. For clustering inputs, the canonical two-column data.frame(`node`, `group`) stored on an existing HTNA model is also accepted. The argument may be omitted when `Nestimate` preserved the partition from an HTNA input. Mutually exclusive with `actor_type`. For an `mcml` input, this optionally overrides `$cluster_members`; otherwise the fitted membership is used.

<code>action</code>	Character. Name of the action/code column. Default "code".
<code>session</code>	Character. Name of the session column. Default "session_id".
<code>order</code>	Character. Name of the within-session order column. Default "order_in_session".
<code>method</code>	Character. Transition method passed to <code>build_network</code> : "relative" (default), "frequency", "co_occurrence", or "attention". Co-occurrence models are undirected and preserve the same actor partition as transition models.
<code>group</code>	Character. Optional name of a column in data used to split sessions into cohorts (e.g. "cluster"). When supplied, one network is built per group level and the result is a named list of htna networks with class c("htna_group", "netobject_group").
<code>disambiguate</code>	Logical. If FALSE (default), the function errors when a code label appears in more than one actor-type group. If TRUE, codes are prefixed with the actor-type label ("Human:Ask", "AI:Ask") so they become distinct nodes.
<code>source_data</code>	Original node-level data for an <code>mcml</code> input. Usually unnecessary when <code>Nestimate::build_mcml()</code> retained its sequence source. Required when the fitted object has no expandable source, such as an <code>mcml</code> built from a transition matrix or edge list. The fitted macro and within-cluster weights are deliberately not stitched together: rebuilding from this source is what preserves cross-cluster transitions.
<code>...</code>	Additional arguments forwarded to <code>build_network</code> .

Value

A `netobject` with the actor partition stored in `cograph`'s canonical schema, so `cograph::plot_htna(net)` auto-detects the groups with no further arguments:

- `$nodes$groups` - actor label per node (the column name `cograph::plot_htna` auto-detects).
- `$node_groups` - data frame with columns `node` and `group` (canonical `cograph_network` schema, also readable by `cograph::get_groups`, `cluster_summary`, and the `print` method for `cograph_network`).
- `$actor_levels` - declared actor ordering, also retained as metadata on `$node_groups` so the table can be passed back to `build_htna()` without losing that order.

All other slots are exactly as returned by `build_network`. Clustering inputs return an `htna_group`; its children retain the actor partition and its outer clustering assignments, posterior probabilities, diagnostics, and other attributes are preserved. An `mcml` input instead returns one `htna`, equivalent to `Nestimate::as_htna()`.

See Also

[build_network](#), [build_tna](#), [plot_htna](#)

Examples

```
data(human_ai, human_simplified, ai_simplified)

# Form 1: named list of per-actor frames
net <- build_htna(list(Human = human_simplified, AI = ai_simplified))
head(net$node_groups)

# Form 2: single combined frame with a row-level actor-type tag
net <- build_htna(human_ai, actor_type = "actor_type")
```

casedrop_reliability_htna

Edge-Weight Case-Dropping Stability

Description

htna-named alias of [Nestimate::casedrop_reliability\(\)](#). Computes the CS-coefficient for the edge-weight vector of a network: the maximum proportion of cases (rows of `x$data`) that can be dropped while the flattened edge-weight vector of the re-estimated network still correlates with the original above threshold in at least certainty of iterations.

Usage

```
casedrop_reliability_htna(x, ...)
```

Arguments

<code>x</code>	An htna network or grouped htna network. See Nestimate::casedrop_reliability() .
<code>...</code>	Additional arguments passed to Nestimate::casedrop_reliability() , such as <code>iter</code> , <code>drop_prop</code> , <code>threshold</code> , <code>certainty</code> , <code>method</code> , <code>include_diag</code> , and <code>seed</code> . See that function for the full, current argument list.

Details

Works on htna networks and grouped htna networks directly.

Suffixed `_htna` to avoid clashing with [Nestimate::casedrop_reliability\(\)](#) when both packages are loaded.

Value

An object of class `net_casedrop_reliability` (single network) or `net_casedrop_reliability_group` (grouped htna). See [Nestimate::casedrop_reliability\(\)](#) for the full component list and the corresponding `plot()` method.

See Also

[reliability_htna\(\)](#), [centrality_stability_htna\(\)](#), [bootstrap_htna\(\)](#).

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
casedrop_reliability_htna(net, iter = 20, seed = 1)
```

centralities_htna

Compute Centrality Measures for an htna Network

Description

Computes a fixed panel of centrality measures via [cograph::cograph](#) and returns them as a tidy data frame: one row per node, one column per measure (plus node, an actor column when the htna partition is set, and a group column when the input is an htna_group).

Usage

```
centralities_htna(
  x,
  measures = c("OutStrength", "InStrength", "ClosenessIn", "ClosenessOut", "Closeness",
    "Betweenness", "BetweennessRSP", "Diffusion", "Clustering"),
  ...
)
```

Arguments

<code>x</code>	An htna network from build_htna() or an htna_group.
<code>measures</code>	Character vector of measure names. Defaults to all nine.
<code>...</code>	Forwarded to the underlying cograph centrality functions.

Details

The measures are: OutStrength, InStrength, ClosenessIn, ClosenessOut, Closeness, Betweenness, BetweennessRSP, Diffusion, Clustering. Path-based measures (closeness and betweenness variants) are computed with `invert_weights = TRUE`, since in transition networks larger weight = stronger link = shorter distance. Strength and the closed-form measures (Diffusion, Clustering) use raw weights.

Mapping from measure name to cograph implementation:

- OutStrength → [cograph::centrality_outstrength\(\)](#)
- InStrength → [cograph::centrality_instrength\(\)](#)
- ClosenessIn → [cograph::centrality_incloseness\(\)](#) (inverted)

- ClosenessOut → `cograph::centrality_outcloseness()` (inverted)
- Closeness → `cograph::centrality_closeness()` (inverted)
- Betweenness → `cograph::centrality_betweenness()` (inverted)
- BetweennessRSP → `cograph::centrality_current_flow_betweenness()`
- Diffusion → `cograph::centrality_diffusion()`
- Clustering → `cograph::centrality_transitivity()`

Value

A data frame with one row per node.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
centralities_htna(net)
```

centrality_stability_htna

Centrality Stability for an htma Network

Description

Thin wrapper around `Nestimate::centrality_stability()` that validates the input is an htma network (or htma_group), forwards the call, and tags the result with an htma_stability class so callers can dispatch on the htma form.

Usage

```
centrality_stability_htna(
  x,
  measures = c("InStrength", "OutStrength", "Betweenness"),
  iter = 1000L,
  drop_prop = seq(0.1, 0.9, by = 0.1),
  threshold = 0.7,
  certainty = 0.95,
  method = "pearson",
  centrality_fn = NULL,
  loops = FALSE,
  seed = NULL
)
```

Arguments

<code>x</code>	An htna network from <code>build_htna()</code> or an <code>htna_group</code> .
<code>measures</code>	Character vector of centrality measures to assess. Default <code>c("InStrength", "OutStrength", "Betweenness")</code> — the three measures whose values are bit-equal between htna's cograph engine and Nestimate's default centrality implementation. To assess htna's nine measures (including the three that differ between engines: <code>BetweennessRSP</code> , <code>Diffusion</code> , <code>Clustering</code>), pass them explicitly together with a custom <code>centrality_fn</code> .
<code>iter</code>	Integer. Number of resamples per drop proportion. Default 1000.
<code>drop_prop</code>	Numeric vector. Proportions of sessions to drop. Default <code>seq(0.1, 0.9, by = 0.1)</code> .
<code>threshold</code>	Numeric in <code>[0, 1]</code> . Correlation threshold for the case-dropping stability coefficient. Default 0.7.
<code>certainty</code>	Numeric in <code>[0, 1]</code> . Probability of meeting the threshold required for a drop proportion to count as stable. Default 0.95.
<code>method</code>	Correlation method, one of "pearson", "spearman", "kendall". Default "pearson".
<code>centrality_fn</code>	Optional function to compute centralities. Default NULL (use Nestimate's internal implementation). See <code>Nestimate::centrality_stability()</code> for the expected signature.
<code>loops</code>	Logical. Whether to retain self-loops. Default FALSE.
<code>seed</code>	Optional integer seed for reproducibility. Default NULL (no seed reset).

Details

For an `htna_group`, the call is iterated per cohort and the result is a named list (one `htna_stability` per cohort), with class `c("htna_stability_group", "list")`.

Value

For a single htna network, an object of class `c("htna_stability", "net_stability")` with the same components as `Nestimate::centrality_stability()`: `cs` (the stability coefficients, one per measure), `correlations` (per-drop-proportion correlation traces), and the parameters that were used. For an `htna_group`, a named list of such objects with class `c("htna_stability_group", "list")`.

See Also

`Nestimate::centrality_stability()`, `bootstrap_htna()`, `reliability_htna()`.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
cs <- centrality_stability_htna(net, iter = 30, seed = 1)
cs$cs
```

certainty_htna

*Closed-form Edge Certainty of a Network***Description**

htna-named wrapper for [Nestimate::certainty\(\)](#). The closed-form counterpart of [bootstrap_htna\(\)](#): rather than resampling, it derives analytic confidence intervals and an inference verdict for each edge weight, which is fast enough to run where a full bootstrap would be costly.

Usage

```
certainty_htna(x, ...)
```

Arguments

<code>x</code>	An htna network from build_htna() (or other object accepted by Nestimate::certainty()).
<code>...</code>	Additional arguments passed to Nestimate::certainty() , such as <code>prior</code> , <code>ci_level</code> , <code>inference</code> , <code>consistency_range</code> , and <code>edge_threshold</code> . See that function for the full, current argument list.

Details

Works on htna networks and grouped htna networks directly.

Suffixed `_htna` to sit alongside [bootstrap_htna\(\)](#) and [reliability_htna\(\)](#) in the htna confirmatory-analysis family.

Value

An object of class `net_certainty` (also inheriting `net_bootstrap`). See [Nestimate::certainty\(\)](#) for the full component list and the corresponding `plot()` method.

See Also

[bootstrap_htna\(\)](#) for the resampling analogue, [reliability_htna\(\)](#), [casedrop_reliability_htna\(\)](#).

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
certainty_htna(net)
```

compare_htna

Compare Heterogeneous Transition Networks

Description

Computes the descriptive network comparison implemented by `Nestimate::compare_model()` while retaining the HTNA actor partition. Comparing two individual networks returns one `htna_comparison`. Passing an `htna_group` as `x` with `y = NULL` compares every cohort pair and returns an `htna_comparison_group`; callers do not need to select or subset cohorts.

Usage

```
compare_htna(x, y = NULL, ...)
```

Arguments

<code>x</code>	An <code>htna</code> network or an <code>htna_group</code> .
<code>y</code>	A second <code>htna</code> network. Must be <code>NULL</code> when <code>x</code> is an <code>htna_group</code> , in which case all cohort pairs are compared.
<code>...</code>	Additional arguments passed to <code>Nestimate::compare_model()</code> , including scaling, measures, and network.

Details

Networks may contain different subsets of the declared nodes. Their weight matrices are aligned to the union of nodes, in canonical actor order, and missing nodes are represented by zero rows and columns. Shared nodes must have the same actor assignment in both networks.

Value

For two networks, an object inheriting from `htna_comparison` and `net_comparison`. In addition to the `Nestimate` comparison fields it contains:

- `node_groups` and `actor_levels`: the aligned actor partition;
- `edge_metrics$source_actor` and `edge_metrics$target_actor`;
- `actor_pair_metrics`: difference summaries for every directed actor-type block;
- `models`: the two original, unmodified HTNA networks.

For an `htna_group`, a named `htna_comparison_group` containing every pairwise comparison is returned.

See Also

`Nestimate::compare_model()`, `plot_htna_diff()`

Examples

```
data(human_ai)
grouped <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
comparisons <- compare_htna(grouped)
names(comparisons)
```

deprune_htna

*Restore All Edges of a Pruned Transition Network***Description**

htna-named wrapper for [Nestimate::net_deprune\(\)](#). Reverses a [prune_htna\(\)](#) step, restoring the original (unpruned) edge set.

Usage

```
deprune_htna(x, ...)
```

Arguments

x	A pruned htna network (or other object accepted by Nestimate::net_deprune()); S3 dispatch on x is preserved.
...	Additional arguments passed to Nestimate::net_deprune() . See that function for details.

Details

The actor partition (`$node_groups`, `$actor_levels`, htna class) is preserved, so the restored network stays htna-aware.

Suffixed `_htna` to avoid clashing with the tna verb `deprune()` and to sit alongside the rest of the htna API.

Value

The depruned htna network. See [Nestimate::net_deprune\(\)](#).

See Also

[prune_htna\(\)](#), [reprune_htna\(\)](#), [pruning_details_htna\(\)](#).

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
pruned <- prune_htna(net, method = "threshold", threshold = 0.05)
deprune_htna(pruned)
```

edge_betweenness_htna *Edge Betweenness Network*

Description

Computes the edge betweenness of every existing edge in an htna network and returns a copy whose `$weights` slot stores those betweenness scores instead of the original transition weights. The actor partition is preserved, so the result can be plotted with `plot_htna()` to visualise which edges carry the most shortest-path traffic.

Usage

```
edge_betweenness_htna(x, directed = TRUE, invert = TRUE)
```

Arguments

<code>x</code>	An htna network from <code>build_htna()</code> .
<code>directed</code>	If TRUE (default), shortest paths follow edge direction.
<code>invert</code>	If TRUE (default), use $1 / \text{weight}$ as the edge cost when computing shortest paths.

Details

Mirrors `tna::betweenness_network()`: edge weights are inverted to costs by default (`invert = TRUE`) – in transition networks a larger transition probability means the edge is "easier" and so the equivalent path cost is smaller.

Value

A copy of `x` whose `$weights` matrix entries are edge-betweenness scores at every position where the original network had a non-zero transition. Class is `c("htna_edge_betweenness", class(x))`.

See Also

`centralities_htna()` for node-level centrality measures, `tna::betweenness_network()` for the tna equivalent.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
eb <- edge_betweenness_htna(net)
plot_htna(eb)           # edge thickness = betweenness score
```

extract_meta_paths	<i>Extract Path Patterns from a Heterogeneous Transition Network</i>
--------------------	--

Description

Discovers recurring patterns in the actor-typed sequences that produced a heterogeneous transition network. Operates at two levels:

Usage

```
extract_meta_paths(
  x,
  level = c("state", "type"),
  length = 2:4,
  schema = NULL,
  type = c("contiguous", "gapped"),
  gap = 1L,
  min_count = 1L,
  min_support = 0,
  min_lift = 0,
  start = NULL,
  end = NULL,
  contain = NULL
)
```

Arguments

x	[htna_network] A network built with build_htna() . Must have \$nodes\$groups, \$node_groups, and \$data populated.
level	[character(1): "state"] "state" returns concrete state-level patterns with a meta_schema rollup column. "type" returns the type-level meta-path summary.
length	[integer(): 2:4] Pattern lengths to enumerate. Ignored when schema is supplied.
schema	[character(1)] Optional. A path template written as elements separated by "->". Each element can be a type name, a concrete state, or "*". See Details.
type	[character(1): "contiguous"] "contiguous" (default) considers consecutive positions; "gapped" considers positions spaced by gap + 1 apart.
gap	[integer(): 1L] Gap size(s) used when type = "gapped". Multiple values produce one row per (length, gap) combination.
min_count	[integer(1): 1L] Minimum total occurrences to retain.

min_support	[numeric(1): 0] Minimum proportion of sequences containing the pattern at least once.
min_lift	[numeric(1): 0] Minimum lift (observed / expected under marginal independence). 0 disables the filter.
start, end, contain	[character()] Filter rows whose first / last element is in start / end, or whose element sequence contains all of contain. Compared against the alphabet of the chosen level (types or concrete states).

Details

- level = "state" (default) - enumerate concrete state-level patterns (e.g. Command->Execute->Command) and annotate each row with the type-level template it instantiates (e.g. Human->AI->Human).
- level = "type" - enumerate type-level meta-paths only (one row per actor-type pattern, summed over its concrete instances).

At either level, a schema can filter the search. Schema parts can be type names (expand to every concrete code in that group), concrete states, or "*" (any element). Parts can be mixed freely - e.g. "Human->Ask->Human" means "any Human code, then Ask, then any Human code". At level = "type" concrete codes resolve to their type, so the same schema becomes Human->AI->Human.

Operates on the actor partition stored on the network by [build_htna\(\)](#).

Value

An object of class `c("htna_meta_paths", "htna_paths", "data.frame")`. At level = "state" the columns are schema, meta_schema, length, gap, count, n_sequences, support, frequency, lift. At level = "type" the meta_schema column is omitted (the schema column already holds the type pattern). Attributes: n_sequences, alphabet, level, and (when supplied) schema.

See Also

[build_htna\(\)](#).

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")

# Concrete state-level patterns of length 2..4 (default)
extract_meta_paths(net)

# Type-level meta-path summary
extract_meta_paths(net, level = "type")

# Concrete instances of a type-level template
extract_meta_paths(net, schema = "Human->AI->Human")

# Mix types, concrete codes, and wildcards
```

```

extract_meta_paths(net, schema = "Human->Ask->Human")
extract_meta_paths(net, schema = "Human->*->Human")

# Gapped patterns; lift threshold
extract_meta_paths(net, length = 3, type = "gapped", gap = 1:2)
extract_meta_paths(net, length = 3, min_lift = 1.2)

```

frequencies_htna	<i>Tidy Per-Actor Frequency Table for an htna Network</i>
------------------	---

Description

Returns the within-network state frequency table, partitioned by actor. One row per (actor, state) pair, with count and proportion columns. The data side of [plot_frequencies_htna\(\)](#).

Usage

```
frequencies_htna(x)
```

Arguments

x An htna network from [build_htna\(\)](#).

Details

Internally a thin wrapper around the htna S3 method shipped by Nestimate (`state_distribution.htna`); exposed under the `frequencies_htna()` name as the canonical "give me the frequency table for this network" entry point.

Suffixed `_htna` to avoid clashing with `Nestimate::frequencies()` (which takes raw long-format data and a column-name spec) when both packages are loaded. If you have raw data rather than an htna network, call `Nestimate::frequencies()` directly.

Value

A data frame with columns group (actor), state, count, proportion.

See Also

[plot_frequencies_htna\(\)](#) for the rendered version, [state_distribution_htna\(\)](#) (same function under the upstream name), [mosaic_plot_htna\(\)](#).

Examples

```

data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
frequencies_htna(net)

```

htna_palette	<i>Default HTNA colour palette</i>
--------------	------------------------------------

Description

A colour palette for up to 6 actor groups, used by `plot_htna()` when no explicit colours are supplied.

Usage

htna_palette

Format

An object of class character of length 6.

Value

A character vector of 6 hex colour codes.

Examples

htna_palette

human_ai	<i>Simplified Human + AI Interaction Sequences</i>
----------	--

Description

A long-format heterogeneous sequence dataset built from `Nestimate::human_long` and `Nestimate::ai_long` by collapsing several near-synonym codes into a smaller, more interpretable alphabet. Suitable as a teaching example for `build_htna()`.

Usage

human_ai

Format

A data frame with 19347 rows and 11 columns:

- message_id** Integer. Source message identifier.
- project** Character. Project label (e.g. "Project_1").
- session_id** Character. Session identifier — pass to `build_htna()` as the session key.
- timestamp** Integer. Unix timestamp of the message.

- session_date** Character. Date of the session (YYYY-MM-DD).
- code** Character. Simplified action code; the code-column value passed to `build_htna()`.
- cluster** Character. Original cluster label from the source data, retained for reference (see Details).
- code_order** Integer. Order of the code within the message.
- order_in_session** Integer. Order of the row within the session — pass as the order key to `build_htna()`.
- actor_type** Character. "AI" or "Human" — the actor partition for `build_htna(actor_type = "actor_type")`.
- phase** Factor with levels "Early" and "Late". Session-level cohort tag from a chronological split: sessions ordered by their first `session_date` (with `session_id` as a deterministic tiebreak), then split in half. Suitable for `build_htna(group = "phase")`.

Details

Code remapping (all other codes pass through unchanged):

- AI: Investigate, Ask, Inquire → Ask; Explain, Report → Report.
- Human: Command, Request → Request; Correct, Verify → Check; Interrupt, Frustrate → Frustrate.

Resulting alphabets:

- AI (6): Ask, Delegate, Execute, Plan, Repair, Report.
- Human (6): Check, Frustrate, Inquire, Refine, Request, Specify.

Because two source codes can map to the same simplified code while originally belonging to different cluster values, a single simplified code may appear with more than one cluster label across rows. The cluster column is preserved verbatim from the source data and should be treated as informational only.

Source

Derived from `Nestimate::human_long` and `Nestimate::ai_long`; see `data-raw/human_ai.R` for the build script.

See Also

`human_simplified`, `ai_simplified`, `human_ai_codebook`.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
plot_htna(net)
```

human_ai_codebook	<i>Code → Actor-Type Codebook for human_ai</i>
-------------------	--

Description

A tidy two-column lookup tagging every simplified code in [human_ai](#) with its actor type. Ready to pass as `node_groups` to `build_htna()` (Form 3b in the vignette("input-formats", package = "htna")).

Usage

```
human_ai_codebook
```

Format

A data frame with 12 rows and 2 columns:

code Character. Simplified action code (one of the 12 Human + AI codes in [human_ai](#)).

actor_type Character. "Human" or "AI".

Source

Derived from [human_simplified](#) and [ai_simplified](#); see `data-raw/human_ai.R` for the build script.

See Also

[human_ai](#), [human_simplified](#), [ai_simplified](#).

Examples

```
data(human_ai, human_ai_codebook)
net <- build_htna(human_ai, node_groups = human_ai_codebook)
plot_htna(net)
```

human_simplified	<i>Simplified Human Interaction Sequences (per-actor frame)</i>
------------------	---

Description

The Human-only slice of [human_ai](#), in a long-format data frame ready to feed the named-list form of `build_htna()`. Codes have already been collapsed into the simplified Human alphabet; see [human_ai](#) for the remapping rules.

Usage

```
human_simplified
```

Format

A data frame with 10796 rows and 10 columns. Schema matches [human_ai](#) minus the phase column; every value in actor_type is "Human".

Source

Derived from `Nestimate::human_long`; see `data-raw/human_ai.R` for the build script.

See Also

[human_ai](#), [ai_simplified](#), [human_ai_codebook](#).

Examples

```
data(human_simplified, ai_simplified)
net <- build_htna(list(Human = human_simplified, AI = ai_simplified))
plot_htna(net)
```

markov_order_test_htna

Markov-Order Adequacy Test

Description

htna-named alias of [Nestimate::markov_order_test\(\)](#). Tests whether a first-order Markov model is adequate for the observed sequences, or whether a higher order is required, by comparing the empirical transition structure against orders `1:max_order` via permutation.

Usage

```
markov_order_test_htna(data, ...)
```

Arguments

data	Raw sequence data: a list of character vectors or a wide-format data frame. See Nestimate::markov_order_test() .
...	Additional arguments passed to Nestimate::markov_order_test() , such as <code>max_order</code> , <code>n_perm</code> , <code>alpha</code> , <code>parallel</code> , <code>n_cores</code> , and <code>seed</code> . See that function for the full, current argument list.

Details

Operates on raw sequence data (a list of character vectors or a wide-format data frame), not on an htna network object. Use alongside [reliability_htna\(\)](#) and [casedrop_reliability_htna\(\)](#) when assessing whether a Markov-1 transition network is appropriate before trusting downstream htna analyses.

Suffixed `_htna` to avoid clashing with `Nestimate::markov_order_test()` when both packages are loaded.

Value

An object of class `markov_order_test` with components `test_table`, `optimal_order`, and the inputs. See [Nestimate::markov_order_test\(\)](#) for full details and the corresponding `plot()` method.

See Also

[reliability_htna\(\)](#), [casedrop_reliability_htna\(\)](#), [centrality_stability_htna\(\)](#).

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
markov_order_test_htna(net$data, max_order = 2, n_perm = 50, seed = 1)
```

mosaic_plot_htna

Chi-square Mosaic Plot of a Transition Network

Description

htna-named alias of [Nestimate::mosaic_plot\(\)](#). Renders a chi-square mosaic where row x column area equals the joint share of (from, to) transitions and fill encodes the standardized residual (blue = over-represented, red = under-represented, white = at-expected).

Usage

```
mosaic_plot_htna(x, ...)
```

Arguments

<code>x</code>	An htna network built with <code>method = "frequency"</code> (or other object accepted by Nestimate::mosaic_plot()); S3 dispatch on <code>x</code> is preserved, so htna objects use <code>mosaic_plot.htna</code> .
<code>...</code>	Additional arguments passed to Nestimate::mosaic_plot() , such as <code>n_perm</code> and <code>seed</code> . See that function for details.

Details

Designed with htna in mind: Nestimate ships an explicit `mosaic_plot.htna` S3 method that recognises the actor partition on htna networks. Pass an htna network from `build_htna()` directly. The underlying transition matrix must be integer-weighted (build with `method = "frequency"`), since the chi-square test needs counts.

Axis tick labels are coloured by actor group using the htna palette (`htna_palette`), matching the colours used elsewhere in htna (e.g. `plot_htna()`).

Suffixed `_htna` to avoid clashing with `Nestimate::mosaic_plot()` when both packages are loaded.

Value

A ggplot or gtable, depending on input shape. See `Nestimate::mosaic_plot()` for full details and the per-class S3 methods.

See Also

`plot_frequencies_htna()` for the marginal-frequency companion view.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type",
                  method = "frequency")
mosaic_plot_htna(net, n_perm = 50, seed = 1)
```

permutation_htna

Permutation Test for Network Differences

Description

htna-named alias of `Nestimate::permutation()`. Tests whether observed edge-weight differences between two networks (or all pairwise differences within a `netobject_group`) exceed what would be expected under a null of identical generating processes.

Usage

```
permutation_htna(x, y = NULL, ...)
```

Arguments

<code>x, y</code>	The two networks to compare, or a single <code>netobject_group</code> in <code>x</code> (with <code>y = NULL</code>) for all pairwise comparisons. See <code>Nestimate::permutation()</code> .
<code>...</code>	Additional arguments passed to <code>Nestimate::permutation()</code> , such as <code>iter</code> , <code>alpha</code> , <code>paired</code> , <code>adjust</code> , <code>measures</code> , <code>nlambda</code> , and <code>seed</code> . See that function for the full, current argument list.

Details

Works on htna networks: the actor partition (`$node_groups`, `$actor_levels`, htna class) is preserved on `result$x / result$y`, so `plot_htna_diff()` can render the result with htna's colour and layout conventions.

Suffixed `_htna` to avoid clashing with `Nestimate::permutation()` when both packages are loaded.

Value

An object of class `net_permutation` (single pair) or `net_permutation_group` (multiple pairs). See `Nestimate::permutation()` for the full slot list.

See Also

`plot_htna_diff()` to plot the result.

Examples

```
data(human_ai)
grp <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
permutation_htna(grp$Early, grp$Late, iter = 50)
```

```
plot.htna_comparison_group
```

Plot All Comparisons in an HTNA Comparison Group

Description

Applies Nestimate's `net_comparison` plot method to every cohort pair.

Usage

```
## S3 method for class 'htna_comparison_group'
plot(x, ...)
```

Arguments

<code>x</code>	An <code>htna_comparison_group</code> from <code>compare_htna()</code> .
<code>...</code>	Passed to the <code>net_comparison</code> plot method, including type and combined.

Value

Invisibly, a named list containing the plot for every comparison.

plot.htna_stability *Plot Method for htna Centrality Stability Objects*

Description

S3 method for plotting htna centrality stability results. Dispatches to cograph's plotting method with htna-specific styling.

Usage

```
## S3 method for class 'htna_stability'
plot(x, ...)
```

Arguments

x An object of class htna_stability from [centrality_stability_htna\(\)](#).
 ... Additional arguments passed to the plotting method.

Value

A ggplot object or plot output, depending on the underlying method.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
cs <- centrality_stability_htna(net, iter = 20, seed = 1)
plot(cs)
```

plot.htna_stability_group *Plot Method for htna Stability Groups*

Description

S3 method for plotting grouped htna stability results.

Usage

```
## S3 method for class 'htna_stability_group'
plot(x, ...)
```

Arguments

x An object of class htna_stability_group from [centrality_stability_htna\(\)](#).
 ... Additional arguments passed to the plotting method.

Value

A combined plot or list of plots for each group.

Examples

```
data(human_ai)
grp <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
cs <- centrality_stability_htna(grp, iter = 20, seed = 1)
plot(cs)
```

plot_centralities	<i>Plot Centrality Measures</i>
-------------------	---------------------------------

Description

Faceted bar plot of node-level centralities, one panel per measure. Mirrors the look of [tna::plot.tna_centralities\(\)](#).

Usage

```
plot_centralities(
  x,
  measures = c("OutStrength", "InStrength", "ClosenessIn", "ClosenessOut", "Closeness",
    "Betweenness", "BetweennessRSP", "Diffusion", "Clustering"),
  by = c("state", "group"),
  reorder = TRUE,
  ncol = 3,
  scales = c("free_x", "fixed"),
  colors = NULL,
  labels = TRUE,
  ...
)
```

Arguments

x	An htna network, htna_group, or htna_centralities data frame from centralities_htna() .
measures	Centralities to plot. Default: all nine.
by	"state" (default) gives each node its own colour; "group" colours by actor group (Human, AI, ...) using htna_palette .
reorder	If TRUE, sort nodes by value within each measure.
ncol	Number of facet columns. Default 3.
scales	Facet scaling: "free_x" (default) or "fixed".
colors	Optional fill colours, recycled per group/node.
labels	If TRUE (default), draw the value next to each bar.
...	Forwarded to centralities_htna() when computing on the fly.

Details

Accepts an htna network, an htna_group, or a data frame produced by `centralities_htna()`. For groups, bars are coloured by group within each panel.

Value

A ggplot object.

See Also

`centralities_htna()`.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
plot_centralities(net)

grp <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
plot_centralities(grp)
```

plot_edge_betweenness_htna

Plot the Edge Betweenness Network

Description

Computes `edge_betweenness_htna()` and renders it with the htna actor styling (multi-actor circular layout, `htna_palette` node colours, actor legend). Edges are scaled by their betweenness score, so the most-traveled shortest-path edges stand out.

Usage

```
plot_edge_betweenness_htna(x, directed = TRUE, invert = TRUE, ...)
```

Arguments

<code>x</code>	An htna network from <code>build_htna()</code> or, equivalently, the result of <code>edge_betweenness_htna()</code> .
<code>directed, invert</code>	Forwarded to <code>edge_betweenness_htna()</code> when <code>x</code> is a plain htna network. Ignored when <code>x</code> already inherits from <code>htna_edge_betweenness</code> .
<code>...</code>	Forwarded to <code>plot_htna()</code> (e.g. <code>minimum</code> , <code>layout</code> , <code>group_colors</code>).

Value

The value returned by `plot_htna()` (invisibly).

See Also

[edge_betweenness_htna\(\)](#), [plot_htna\(\)](#).

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
plot_edge_betweenness_htna(net)
plot_edge_betweenness_htna(net, minimum = 5)
```

plot_frequencies_htna *Plot State Frequencies (htna-named)*

Description

Renders one of three views of an htna network's state frequencies: the upstream treemap (default), a combined actor-coloured bar chart, or a per-actor faceted bar chart.

Usage

```
plot_frequencies_htna(x, view = c("treemap", "bars", "facet"), ...)
```

Arguments

x	An htna network from build_htna() .
view	One of "treemap" (default), "bars", or "facet". • "treemap" forwards to Nestimate::plot_state_frequencies() and renders the chart automatically; returns the underlying state_freq object invisibly. • "bars" builds a combined bar chart with all states on one y-axis, sorted by count, fill coloured by actor (using htna_palette keyed off x\$actor_levels). Returns the ggplot. • "facet" builds a per-actor faceted bar chart with scales = "free_y" so each panel only shows its own actor's states. Returns the ggplot.
...	Forwarded to Nestimate::plot_state_frequencies() when view = "treemap". Ignored for "bars" and "facet" — those return ggplot objects, so customisation works through standard ggplot composition (+ theme(...), + labs(...), + scale_fill_*(...), etc.).

Details

Suffixed _htna to avoid clashing with [Nestimate::plot_state_frequencies\(\)](#) when both packages are loaded.

Value

For `view = "treemap"`: the `state_freq` object invisibly. For `"bars" / "facet"`: a `ggplot`, returned visibly so the chart auto-prints and `standard + composition` works.

See Also

[frequencies_htna\(\)](#) for the underlying tidy table, [state_frequencies_htna\(\)](#), [state_distribution_htna\(\)](#), [mosaic_plot_htna\(\)](#).

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
plot_frequencies_htna(net, view = "treemap")
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot_frequencies_htna(net, view = "bars")
}
```

plot_htna

Plot a Heterogeneous Transition Network

Description

Wrapper around `cograph::plot_htna()` with defaults suited for HTNA networks built by `build_htna()`. When `layout = "circular"`, a custom layout is computed so that the first actor group appears on the left.

Usage

```
plot_htna(
  x,
  layout = "circular",
  group_colors = htna_palette,
  group_shapes = htna_shape_palette,
  minimum = 0.05,
  ...
)

## S3 method for class 'htna'
plot(x, ...)

## S3 method for class 'htna_group'
plot(x, ...)
```

Arguments

x	A network object produced by <code>build_htna()</code> .
layout	Character. Layout algorithm. Default "circular".
group_colors	Character vector of colours, one per actor group. Defaults to the built-in <code>htna_palette</code> .
group_shapes	Character vector of node shapes, one per actor group. Defaults to the built-in <code>htna_shape_palette</code> .
minimum	Numeric. Minimum absolute edge weight to display. Edges below this threshold are hidden. Default 0.05.
...	Additional arguments passed to <code>cograph::plot_htna()</code> .

Value

Called for its side effect (a plot). Returns x invisibly.

See Also

`cograph::plot_htna()`, `build_htna()`

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
plot_htna(net)
plot_htna(net, layout = "auto", minimum = 0.1)
```

plot_htna_bootstrap	<i>Plot an htna Bootstrap Result</i>
---------------------	--------------------------------------

Description

Renders the bootstrap result via `cograph::splot.net_bootstrap()` so the CI / significance / dashed-edge overlay is preserved, but overrides the layout and node styling to match `plot_htna()` (multi-group circular layout, warm `htna_palette`, darkened donut).

Usage

```
plot_htna_bootstrap(boot, group_colors = htna_palette, ...)

## S3 method for class 'htna_bootstrap'
plot(x, ...)
```

Arguments

boot	An object from <code>bootstrap_htna()</code> (or a <code>htna_bootstrap_group</code> list of them).
group_colors	Character vector of colours, one per actor group. Defaults to the built-in <code>htna_palette</code> .
...	Forwarded to <code>cograph::splot.net_bootstrap()</code> (e.g. <code>display</code> , <code>show_ci</code> , <code>show_stars</code>). User args win.
x	Same as <code>boot</code> ; used when calling via the <code>plot()</code> generic.

Details

For an `htna_bootstrap_group` (i.e. the result of running `bootstrap_htna()` on a `build_htna(..., group = ...)` output), each element is plotted individually with its name as the title - this function does not manage `par(mfrow=...)`.

Value

The value returned by `cograph::splot.net_bootstrap()` (invisibly), or the input group invisibly.

See Also

`bootstrap_htna()`, `plot_htna()`, `cograph::splot.net_bootstrap()`.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
boot <- bootstrap_htna(net, iter = 50)
plot_htna_bootstrap(boot)
plot_htna_bootstrap(boot, display = "significant")
```

plot_htna_diff

Plot the Difference Between Two htna Networks

Description

Renders the elementwise edge-weight difference $x - y$ as a heterogeneous transition network. Accepts:

Usage

```
plot_htna_diff(
  x,
  y = NULL,
  pos_color = "#009900",
  neg_color = "#C62828",
  group_colors = htna_palette,
  ...
)
```

Arguments

<code>x</code>	One of: an htna network, a <code>net_permutation</code> result, or a <code>net_permutation_group</code> .
<code>y</code>	htna network (only used when <code>x</code> is also an htna network).
<code>pos_color</code>	Edge colour for positive differences. Default <code>"#009900"</code> .
<code>neg_color</code>	Edge colour for negative differences. Default <code>"#C62828"</code> .
<code>group_colors</code>	Character vector of colours, one per actor group. Defaults to the built-in htna_palette .
<code>...</code>	Forwarded to the underlying splot dispatcher (<code>cograph::splot()</code> or <code>cograph::splot.net_permutation()</code>).

Details

- Two htna networks (`x`, `y`): plots `x - y` via `cograph::splot()` with positive/negative edge coloring.
- A `net_permutation` result: routes through `cograph::splot.net_permutation()` so the significance overlay (significant pos/neg edges, dashed non-significant edges, stars) is drawn upstream.
- A `net_permutation_group`: iterates over all pairwise comparisons and plots each (no `par(mfrow=...)` management - user controls layout).

In all cases the layout, node colours and donut match `plot_htna()`.

Value

The value of the underlying splot call (invisibly).

See Also

`plot_htna()`, `build_htna()`, `Nestimate::permutation()`.

Examples

```
data(human_ai)
grp <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
plot_htna_diff(grp$Early, grp$Late)

perm <- permutation_htna(grp$Early, grp$Late, iter = 50)
plot_htna_diff(perm)
plot_htna_diff(perm, show_nonsig = TRUE)
```

plot_sequences	<i>Sequence plot generic</i>
----------------	------------------------------

Description

S3 generic dispatched on x. Calling `plot_sequences(net)` on an htna network forwards to [sequence_plot_htna\(\)](#).

Usage

```
plot_sequences(x, ...)
```

Arguments

x	An object to plot.
...	Forwarded to the method.

Value

Method-defined.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
plot_sequences(net)
```

print.htna_stability	<i>Print Method for htna Centrality Stability Objects</i>
----------------------	---

Description

S3 method for printing htna centrality stability results.

Usage

```
## S3 method for class 'htna_stability'
print(x, ...)
```

Arguments

x	An object of class <code>htna_stability</code> from centrality_stability_htna() .
...	Additional arguments passed to the print method.

Value

Invisibly returns the object, prints stability information.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
cs <- centrality_stability_htna(net, iter = 20, seed = 1)
print(cs)
```

```
print.htna_stability_group
```

Print Method for htna Stability Groups

Description

S3 method for printing grouped htna stability results.

Usage

```
## S3 method for class 'htna_stability_group'
print(x, ...)
```

Arguments

<code>x</code>	An object of class <code>htna_stability_group</code> from <code>centrality_stability_htna()</code> .
<code>...</code>	Additional arguments passed to the print method.

Value

Invisibly returns the object, prints group stability information.

Examples

```
data(human_ai)
grp <- build_htna(human_ai, actor_type = "actor_type", group = "phase")
cs <- centrality_stability_htna(grp, iter = 20, seed = 1)
print(cs)
```

prune_htna

*Prune Weak Edges from a Transition Network***Description**

htna-named wrapper for [Nestimate::net_prune\(\)](#). Removes edges that fall below a weight/threshold rule, returning a pruned network of the same shape.

Usage

```
prune_htna(x, ...)
```

Arguments

<code>x</code>	An htna network from build_htna() (or other object accepted by Nestimate::net_prune()); S3 dispatch on <code>x</code> is preserved.
<code>...</code>	Additional arguments passed to Nestimate::net_prune() , such as <code>method</code> , <code>threshold</code> , <code>lowest</code> , <code>level</code> , and <code>boot</code> . See that function for the full, current argument list.

Details

The actor partition (`$node_groups`, `$actor_levels`, htna class) is preserved on the pruned network, so [plot_htna\(\)](#) and the other htna verbs keep working on the result.

Suffixed `_htna` to avoid clashing with the tna verb `prune()` and to sit alongside the rest of the htna API.

Value

A pruned htna network. See [Nestimate::net_prune\(\)](#) for details.

See Also

[deprune_htna\(\)](#), [reprune_htna\(\)](#), [pruning_details_htna\(\)](#), [build_htna\(\)](#).

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
prune_htna(net, method = "threshold", threshold = 0.05)
```

pruning_details_htna *Tidy Report of a Network's Pruning*

Description

htna-named wrapper for `Nestimate::net_pruning_details()`. Returns a tidy data frame describing which edges were kept or removed by a `prune_htna()` step and why.

Usage

```
pruning_details_htna(x, ...)
```

Arguments

<code>x</code>	A pruned htna network (or other object accepted by <code>Nestimate::net_pruning_details()</code>); S3 dispatch on <code>x</code> is preserved.
<code>...</code>	Additional arguments passed to <code>Nestimate::net_pruning_details()</code> . See that function for details.

Details

Suffixed `_htna` to avoid clashing with the tna verb `pruning_details()` and to sit alongside the rest of the htna API.

Value

A data frame, one row per edge. See `Nestimate::net_pruning_details()` for details.

See Also

`prune_htna()`, `deprune_htna()`, `reprune_htna()`.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
pruned <- prune_htna(net, method = "threshold", threshold = 0.05)
pruning_details_htna(pruned)
```

reliability_htna	<i>Network Reliability for an htna Network</i>
------------------	--

Description

Thin wrapper around `Nestimate::network_reliability()` that preserves the htna actor partition on every model in the returned `$models` slot, so downstream htna-aware code (plotting, centralities, etc.) keeps working on the reliability output.

Usage

```
reliability_htna(..., iter = 1000L, split = 0.5, scale = "none", seed = NULL)
```

Arguments

<code>...</code>	One or more htna networks built by <code>build_htna()</code> .
<code>iter</code>	Integer. Number of split-half iterations. Default 1000.
<code>split</code>	Numeric in (0, 1). Proportion of sessions used for the first half-network. Default 0.5.
<code>scale</code>	One of "none", "minmax", "standardize", "proportion". Forwarded to <code>Nestimate::network_reliability()</code> .
<code>seed</code>	Optional integer seed for reproducibility. Default NULL (no seed reset).

Details

Mirrors the signature of the underlying function: pass one or more networks via `...` plus the optional `iter`, `split`, `scale`, and `seed` arguments. All networks passed through `...` must be htna networks built by `build_htna()`.

Value

An object of class `c("htna_reliability", "net_reliability")`, with the same components as `Nestimate::network_reliability()` — `iterations`, `summary`, `models`, `iter`, `split`, `scale` — and each entry of `$models` carrying the htna actor partition (`$nodes$groups`, `$node_groups`, `$actor_levels`).

See Also

`Nestimate::network_reliability()`, `bootstrap_htna()`.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
rel <- reliability_htna(net, iter = 30, seed = 1)
rel$summary
```

reprune_htna

*Re-apply a Previous Pruning Rule***Description**

htna-named wrapper for [Nestimate::net_reprune\(\)](#). Re-applies the pruning rule recorded on a network (e.g. after the weights have been re-estimated), without having to restate the rule.

Usage

```
reprune_htna(x, ...)
```

Arguments

<code>x</code>	An htna network carrying a prior pruning rule (or other object accepted by Nestimate::net_reprune()); S3 dispatch on <code>x</code> is preserved.
<code>...</code>	Additional arguments passed to Nestimate::net_reprune() . See that function for details.

Details

The actor partition (`$node_groups`, `$actor_levels`, htna class) is preserved, so the re-pruned network stays htna-aware.

Suffixed `_htna` to avoid clashing with the tna verb `reprune()` and to sit alongside the rest of the htna API.

Value

The re-pruned htna network. See [Nestimate::net_reprune\(\)](#).

See Also

[prune_htna\(\)](#), [deprune_htna\(\)](#), [pruning_details_htna\(\)](#).

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
pruned <- prune_htna(net, method = "threshold", threshold = 0.05)
reprune_htna(deprune_htna(pruned))
```

sequence_compare_htna *Subsequence Pattern Comparison Across Groups*

Description

Extends `Nestimate::sequence_compare()` with a `level` argument that lets the comparison run on type-level (meta-path) sequences as well as state-level sequences.

Usage

```
sequence_compare_htna(x, group = NULL, level = c("state", "type"), ...)
```

Arguments

<code>x</code>	A grouped htna network, a single htna network (paired with <code>group</code>), or wide-format sequence data. See <code>Nestimate::sequence_compare()</code> .
<code>group</code>	Grouping specification for the cohorts to compare, when not already carried by a grouped htna network. See <code>Nestimate::sequence_compare()</code> .
<code>level</code>	Character. "state" (default) compares concrete state-level k-grams; "type" first replaces each state with its actor type so the comparison runs on meta-paths (e.g. Human->AI->Human). Only meaningful when <code>x</code> is an htna network with an actor partition.
<code>...</code>	Additional arguments passed to <code>Nestimate::sequence_compare()</code> , such as <code>sub</code> , <code>min_freq</code> , <code>test</code> , <code>iter</code> , and <code>adjust</code> . See that function for the full, current argument list.

Details

Extracts all k-gram patterns (subsequences of length `k`) from the sequences in each cohort, computes standardised residuals against the independence model, and optionally runs a permutation or chi-square test for differences in pattern rates between cohorts.

Operates on grouped htna networks (built via `build_htna(..., group = ...)`), single htna networks paired with a `group` argument, or wide-format sequence data with an explicit `group` argument. The actor partition itself is not consumed when `level = "state"` — sequence comparison is between cohorts of sessions, not between actors. When `level = "type"`, concrete codes are folded into their actor type before pattern enumeration, so the comparison runs on meta-paths.

Value

An object of class `net_sequence_compare`. See `Nestimate::sequence_compare()` for full details and the corresponding `plot()` method.

See Also

`permutation_htna()` for whole-network differences, `mosaic_plot_htna()` for single-step transition residuals, `extract_meta_paths()` for descriptive meta-path enumeration.

Examples

```
data(human_ai)
grp <- build_htna(human_ai, actor_type = "actor_type", group = "phase")

# State-level comparison (default)
sequence_compare_htna(grp, iter = 50)

# Meta-path comparison
sequence_compare_htna(grp, level = "type", iter = 50)
```

sequence_plot_htna	<i>Plot Per-Actor Sequences From an htna Network</i>
--------------------	--

Description

Extracts each actor's events from the combined wide sequence in `net$data` (using `net$node_groups` as the node-to-actor lookup), compresses each session into the actor's own ordered events, pads to a common width, and renders with `Nestimate::sequence_plot()` grouped by actor. Each session contributes one row per actor that had at least one event in it.

Usage

```
sequence_plot_htna(
  net,
  by = c("state", "group"),
  type = c("index", "heatmap", "distribution"),
  grouped_legend = TRUE,
  ...
)

## S3 method for class 'htna'
plot_sequences(x, ...)
```

Arguments

<code>net</code>	An htna network from <code>build_htna()</code> . Must have <code>\$data</code> and <code>\$node_groups</code> populated.
<code>by</code>	"state" (default) keeps state-level colouring with one row per (session, actor) extracted from <code>net\$data</code> ; "group" re-colours the original combined session matrix by actor (each cell = the actor that acted at that time step).
<code>type</code>	Sequence plot layout: "index" (default; one panel per actor, vertically stacked), "heatmap" (single carpet with a white separator at the actor boundary, controllable via <code>k_line_width</code>), or "distribution" (one stacked-area panel per actor).
<code>grouped_legend</code>	Logical. If TRUE (default) and <code>by = "state"</code> , the per-state legend is split into one block per actor with the actor name as a sub-title.

... Forwarded to `Nestimate::sequence_plot()`.
 x Same as `net`; used when calling via the `plot_sequences()` generic.

Value

Invisibly, the list returned by `Nestimate::sequence_plot()`.

See Also

`Nestimate::sequence_plot()`, `build_htna()`.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")

sequence_plot_htna(net)                # index, faceted
sequence_plot_htna(net, type = "heatmap") # single carpet, white gulf
sequence_plot_htna(net, type = "distribution")
```

state_distribution_htna

State Distribution Across Time

Description

htna-named alias of `Nestimate::state_distribution()`. Returns the per-timestep distribution of states across sequences, suitable for driving stacked-area or bar plots.

Usage

```
state_distribution_htna(x, ...)
```

Arguments

x An htna network (or other object accepted by `Nestimate::state_distribution()`); S3 dispatch on `x` is preserved, so htna objects use `state_distribution.htna`.
 ... Additional arguments passed to `Nestimate::state_distribution()`. See that function for details.

Details

Designed with htna in mind: Nestimate ships an explicit `state_distribution.htna` S3 method that uses the actor partition carried by `build_htna()` networks.

Suffixed `_htna` to avoid clashing with `Nestimate::state_distribution()` when both packages are loaded.

Value

A data frame with one row per (timestep, state). See `Nestimate::state_distribution()` for full details.

See Also

`state_frequencies_htna()` for the within-network summary.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
state_distribution_htna(net)
```

state_frequencies_htna

Tidy State Frequency Table

Description

htna-named alias of `Nestimate::state_frequencies()`. Returns a tidy data frame with state, count, and proportion columns summarising the within-network state vocabulary.

Usage

```
state_frequencies_htna(data, ...)
```

Arguments

data	Raw sequence data summarised into a state-frequency table. See <code>Nestimate::state_frequencies()</code> .
...	Additional arguments passed to <code>Nestimate::state_frequencies()</code> . See that function for details.

Details

Companion to `plot_frequencies_htna()` (which is htna-aware via an explicit S3 method). `state_frequencies_htna()` itself operates on the raw sequence data and is the data side of the same family used by the htna tutorials.

Suffixed `_htna` to avoid clashing with `Nestimate::state_frequencies()` when both packages are loaded.

Value

A data frame. See `Nestimate::state_frequencies()` for details.

See Also

[plot_frequencies_htna\(\)](#), [state_distribution_htna\(\)](#).

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
state_frequencies_htna(net$data)
```

summary.htna	<i>Summarise a Heterogeneous Transition Network</i>
--------------	---

Description

Prints a per-actor summary of an htna network: which nodes belong to which actor type and how the non-zero edges distribute across the actor partition. The result is also returned invisibly as a list so callers can inspect the structured summary programmatically.

Usage

```
## S3 method for class 'htna'
summary(object, max_nodes = 12L, ...)

## S3 method for class 'htna_group'
summary(object, max_nodes = 12L, ...)
```

Arguments

object	An htna network from build_htna() or an htna_group.
max_nodes	Integer. Maximum number of nodes to list per actor type before truncating with an ellipsis. Default 12.
...	Forwarded for compatibility; currently unused.

Value

Invisibly, a list with components:

- `actors` - data frame with one row per actor type (`actor`, `n_nodes`, `nodes`).
- `edges_by_actor` - integer matrix of non-zero edge counts, rows are source actor, columns are target actor.
- `network_metrics` - standard network-level descriptive metrics from the Nestimate engine (node/edge counts, density, distance, strength, degree centralization, and reciprocity).
- `n_nodes`, `n_edges`, `n_sessions`, `n_timesteps`, `method`.

Examples

```
data(human_ai)
net <- build_htna(human_ai, actor_type = "actor_type")
summary(net)
```

Index

* datasets

- ai_simplified, [3](#)
- htna_palette, [22](#)
- human_ai, [22](#)
- human_ai_codebook, [24](#)
- human_simplified, [24](#)

- ai_simplified, [3](#), [23–25](#)
- as.igraph.htna, [4](#)
- as.igraph.htna_group(as.igraph.htna), [4](#)
- association_rules_htna, [5](#)

- bayes_compare_htna, [6](#)
- bootstrap, [7](#)
- bootstrap.htna(bootstrap.htna), [7](#)
- bootstrap_htna, [7](#)
- bootstrap_htna(), [7](#), [12](#), [14](#), [15](#), [35](#), [41](#)
- build_htna, [3](#), [8](#), [22–24](#)
- build_htna(), [8](#), [12](#), [14](#), [15](#), [18–21](#), [27](#),
[31–34](#), [36](#), [39](#), [41](#), [44](#), [45](#), [47](#)
- build_network, [9–11](#)
- build_tna, [11](#)

- casedrop_reliability_htna, [11](#)
- casedrop_reliability_htna(), [15](#), [26](#)
- centralities_htna, [12](#)
- centralities_htna(), [18](#), [30](#), [31](#)
- centrality_stability_htna, [13](#)
- centrality_stability_htna(), [12](#), [26](#), [29](#),
[37](#), [38](#)
- certainty_htna, [15](#)
- cograph::centrality_betweenness(), [13](#)
- cograph::centrality_closeness(), [13](#)
- cograph::centrality_current_flow_betweenness(), [13](#)
- cograph::centrality_diffusion(), [13](#)
- cograph::centrality_incloseness(), [12](#)
- cograph::centrality_instrength(), [12](#)
- cograph::centrality_outcloseness(), [13](#)
- cograph::centrality_outstrength(), [12](#)

- cograph::centrality_transitivity(), [13](#)
- cograph::cograph, [12](#)
- cograph::plot_htna(), [33](#), [34](#)
- cograph::splot(), [36](#)
- cograph::splot.net_bootstrap(), [34](#), [35](#)
- cograph::splot.net_permutation(), [36](#)
- compare_htna, [16](#)
- compare_htna(), [28](#)

- deprune_htna, [17](#)
- deprune_htna(), [39](#), [40](#), [42](#)

- edge_betweenness_htna, [18](#)
- edge_betweenness_htna(), [31](#), [32](#)
- extract_meta_paths, [19](#)
- extract_meta_paths(), [43](#)

- frequencies_htna, [21](#)
- frequencies_htna(), [5](#), [33](#)

- htna_palette, [22](#), [27](#), [30](#), [31](#), [34–36](#)
- human_ai, [3](#), [22](#), [24](#), [25](#)
- human_ai_codebook, [3](#), [23](#), [24](#), [25](#)
- human_simplified, [3](#), [23](#), [24](#), [24](#)

- igraph::graph_from_adjacency_matrix(),
[4](#)
- igraph::igraph, [4](#)

- markov_order_test_htna, [25](#)
- mosaic_plot_htna, [26](#)
- mosaic_plot_htna(), [5](#), [21](#), [33](#), [43](#)

- Nestimate::association_rules(), [5](#)
- Nestimate::bayes_compare(), [6](#)
- Nestimate::bootstrap_network(), [7](#), [8](#)
- Nestimate::build_network(), [5](#)
- Nestimate::casedrop_reliability(), [11](#)
- Nestimate::centrality_stability(), [13](#),
[14](#)
- Nestimate::certainty(), [15](#)

Nestimate::compare_model(), 16
 Nestimate::markov_order_test(), 25, 26
 Nestimate::mosaic_plot(), 26, 27
 Nestimate::net_deprune(), 17
 Nestimate::net_prune(), 39
 Nestimate::net_pruning_details(), 40
 Nestimate::net_reprune(), 42
 Nestimate::network_reliability(), 41
 Nestimate::permutation(), 27, 28, 36
 Nestimate::plot_state_frequencies(), 32
 Nestimate::sequence_compare(), 43
 Nestimate::sequence_plot(), 44, 45
 Nestimate::state_distribution(), 45, 46
 Nestimate::state_frequencies(), 46

 permutation_htna, 27
 permutation_htna(), 6, 43
 plot.htna (plot_htna), 33
 plot.htna_bootstrap
 (plot_htna_bootstrap), 34
 plot.htna_comparison_group, 28
 plot.htna_group (plot_htna), 33
 plot.htna_stability, 29
 plot.htna_stability_group, 29
 plot_centralities, 30
 plot_edge_betweenness_htna, 31
 plot_frequencies_htna, 32
 plot_frequencies_htna(), 21, 27, 46, 47
 plot_htna, 11, 33
 plot_htna(), 18, 22, 27, 31, 32, 34–36, 39
 plot_htna_bootstrap, 34
 plot_htna_diff, 35
 plot_htna_diff(), 6, 16, 28
 plot_sequences, 37
 plot_sequences.htna
 (sequence_plot_htna), 44
 print.htna_stability, 37
 print.htna_stability_group, 38
 prune_htna, 39
 prune_htna(), 17, 40, 42
 pruning_details_htna, 40
 pruning_details_htna(), 17, 39, 42

 reliability_htna, 41
 reliability_htna(), 12, 14, 15, 26
 reprune_htna, 42
 reprune_htna(), 17, 39, 40

 sequence_compare_htna, 43
 sequence_plot_htna, 44
 sequence_plot_htna(), 37
 state_distribution_htna, 45
 state_distribution_htna(), 21, 33, 47
 state_frequencies_htna, 46
 state_frequencies_htna(), 33, 46
 summary.htna, 47
 summary.htna_group (summary.htna), 47

 tna::betweenness_network(), 18
 tna::plot.tna_centralities(), 30